AD-A037 190 ARIZONA UNIV TUCSON DEPT OF CHEMISTRY F/G 9/2
THREADED CODE FOR LABORATORY COMPUTERS.(U)
FEB 77 J B PHILLIPS, M F BURKE, G S WILSON N00014-75-C-0512
UNCLASSIFIED NL

| OF |
AD
A037190

END
DATE
FILMED
4-77

ADA037190

OFFICE OF NAVAL RESEARCH
Contract N00014-75-C-0512
Task No. NR 051-518
Technical Report No. 8

12

THREADED CODE FOR LABORATORY COMPUTERS

J. B. Phillips, M. F. Burke and G. S. Wilson

Chemistry Department
University of Arizona
Tucson, Arizona 85721

DDC
RECEIVED
MAR 18 1977
C

Submitted for Publication in

Communications of the ACM

February 1977

Reproduction in whole or in part is permitted for any purpose of the United States Government

Approved for Public Release: Distribution Unlimited

COPY AVAILABLE TO DDC DOES NOT PERMIT FULLY LEGIBLE PRODUCTION

Abstract

A typical minicomputer has been transformed into a threaded code machine by the addition of a simple, fast interpreter implemented in microcode. When made directly available to the programmer, the threaded code programming technique is a very convenient and efficient means of structuring programs, particularly in systems where programs are continually being modified.

Key Words and Phrases:

interpreter, threaded code, machine code, subroutine calls, time tradeoff, space tradeoff, compiled code, code generation, laboratory computers, problem oriented languages

CR Categories 4.12, 4.13, 4.21, 4.22

Acknowledgement

This work was supported in part by Office of Naval Research through ONR Contract No. N00014-75-C-0512.

ACCESSION for
NTIS White Section [✓]
DDC Buff Section ☐
UNANNOUNCED ☐
JUSTIFICATION ....
BY ....
DISTRIBUTION/AVAILABILITY CODES
DIST. AVAIL. and/or SPECIAL
A

I. Introduction

In the course of developing a programming and operating system for a laboratory computer network, an interpretive technique related to the threaded code (1,3) programming technique has been developed. In threaded code a program consists of a string of addresses pointing to service routines. The program is executed interpretively as shown in Figure 1. Upon completion of each service routine, control is transferred to the next one in sequence by incrementing the threaded code position counter and then jumping indirectly through the address in the threaded program to the next service routine. This process can be quite efficient. In fact, for some computer instruction sets (e.g. the PDP 11) it is faster than the standard subroutine call and return. Thus threaded code and related techniques offer most of the advangages of an interpretive mode of operation without the major penalty of slow program execution.

A programming system based on the concept of threaded code, FORTH (4), has been implemented on a number of different computers used in instrument control applications. It uses the idea of indirect threaded code (3) for greater flexibility in defining service routines. Specifically, service routines are provided to move the threaded code position counter to and from a stack so that one threaded program may call another as shown in Figure 2. This capability is an extremely important extention of the threaded code technique because it allows a very convenient hierarchical structuring of programs. In addition, FORTH provides for interactive compilation of both threaded code and assembly level service routines. This particular programming technique is limited in two ways for our approach to laboratory computers: (1) dedication to single user; (2) a syntax which is difficult to use in larger programs.

Threaded code can be used to produce highly hierarchically structured programs which, in addition to being smaller in size than equivalent conventionally structured programs, are also often easier to write and understand. This is closely related to the idea of structured programming (2) in which one of the basic principles is the liberal use of subprogram structures (7). Because of its potential usefulness as an efficient means of structuring programs, it is important that the threaded programming technique be further developed to make it more generally applicable. Improvements can be made both in the design of the technique itself and in methods of writing structured programs using it.

II. Microcode Implementation of a Threaded Code System

HISS, Hierarchical Interactive Sharing System, an operating system written using threaded code, has been implemented for a Hewlett Packard 2100 with the writable control store (WCS) option installed. It is an experimental system for which efficiency in software implementation and maintenance are more important than efficiency in operation. For this reason, it was decided to code as much of it as possible in an interpretive high level language rather than assembly language. New microcode stored in WCS was used to extend the instruction set for interpretation of threaded code. Operating systems have been previously implemented in an interpretive fashion, but with substaintial execution speed penalties (6).

Basically, the flow of control is as illustrated in Figure 2. The threaded code interpreter is similar to the one described by Bell (1) and works in the following way:

Step 1.	Increment PC on top of control stack.
Step 2.	Fetch S from PCth address of memory.
Step 3(a)	If S denotes a service routine, execute it.
Step 3(b)	If not, push the address of S to control stack.
Step 4.	Go to Step 1.

A service routine, NEXT, is provided to delete the value of PC on top of the control stack returning control to the calling threaded program.

The threaded code interpreter and its associated routines use 241_8 of the 400_8 words available in one WCS module. The rest of the WCS space is available for use by low level service routines or user defined instructions. An average interpreter cycle takes about 12 μsec. of overhead time. Of this, about 4 μsec. is used in fetching the instruction from a microcode implemented virtual memory which is useful in this particular system but is not required by the threaded programming technique itself.

Without microcode, threaded code would not be very efficient on the HP 2100 because the standard instruction set does not include any index instructions or registers for implementing stack operations. Many other conventional instruction sets would be better than the HP 2100 for this application, but with the HP 2100 microprogramming capability a threaded code interpreter can be implemented almost as efficiently as if the computer was designed specifically for it. Using microcode the above threaded code interpreter, including the NEXT routine, can be implemented as a single instruction which is executed at the end of each service routine.

A threaded program in the HISS system consists of a string of code numbers. This technique is essentially equivalent to the strings of addresses used in previous threaded code systems. The instruction required to transfer control between service routines is more specialized than a simple jump instruction, but through the use of micorcode it can be implemented almost as efficiently.

There are several reasons for this use of code numbers instead of addresses. First, some service routines are implemented in microcode and therefore do not have core addresses. Second, the microcoded interpreter can index a table of addresses in core just as fast as it can fetch an indirect address for the indirect threaded code technique (3). Third, the routine to push threaded code addresses to the control stack, Step 3(b), should be part of the microcoded interpreter instead of an indirectly referenced service routine as in FORTH. It is simpler and more efficient to make the Step 3 decision based on the range of the code number rather than the value of a word fetched from the program. Finally, instead of the full 16 bit words required for addresses, 8 bits are sufficient to specify most code numbers significantly reducing the memory required for threaded code.

The utility of this structure can best be illustrated by analogy with the concept of a computer's instruction set. Almost always an instruction set includes a subprogram calling instruction which essentially allows the programmer to define new instructions by writing programs using the basic instruction set and any previously defined subprograms. The threaded programming technique is a generalization of the hierarchical subprogram idea. Instead of one instruction code for calling subprograms inside a low level instruction set, all but a very few codes are used to call subprograms and the general purpose low level instruction set is confined to a special mode of operation. A hierarchical set of subprograms thus becomes the instruction set for a special purpose system. The set of instructions provided by lower levels are more appropriate to a specific problem than a general purpose language would be, so higher level programs can be made shorter and more understandable.

Different instruction sets can be defined for different applications. In fact, through the use of local codes the instruction set can vary in different parts of a program. By localizing the range of instruction codes very efficient specialized instructions can be defined and used in one part of a program. The same instruction codes are then reused in other parts of the program. Since a given local instruction has a specific range from which it is callable, the programmer implementing it need not worry about making it completely foolproof and can so produce simpler, more efficient code. Also, a programmer working at higher levels is not tempted to use a specialized lower level instruction which was not intended for his use. Contrast this with the FORTH system in which, generally, all lower level instructions are useable by anyone.

Codes less than 20_8 are interpreted as service routines at Step 3(a). All other codes are used to index into tables of threaded program addresses which are pushed to the control stack at Step 3(b). Control is transferred to one of the 20_8 microcoded service routines via an indexed microcode jump table. Most of these routines simply perform some operation which is important enough to be in microcode, but three of them (NEXT, XO and ASOP) are central to the operation of this threaded code system. NEXT deletes the top address from the control stack, and as a result, transfers control from the current threaded code routine to the calling threaded program whose address is next on the control stack. XO uses the immediately following 8 bit byte to index into a table of addresses of service routines in core. These service routines provide basic arithmetic and an interface with the core resident operating system. ASOP switches the computer from the threaded code interpreter mode of operation to its regular machine code mode. The machine code instructions follow immediately after the ASOP code.

Parameters are passed to and results returned from service routines and sub threaded programs through the use of a data stack. This data stack is distinct from and should not be confused with the control stack which contains the return addresses for sub threaded programs. Several of the basic microcoded service routines provide for elementary operations on the data stack including pushing constants to it, deleting the top word, and duplicating the top word. Many other data stack operations such as simple arithmetic operations are implemented as service routines through the XO code.

Additional features provided by this threaded code system include virtual memory, locally defined instruction codes, and exceptional condition exits. These are important for the overall usefulness of the system but are not necessary for understanding the principles of threaded code and so will not be discussed here.

III. Utility of Threaded Code for Laboratory Computers

The biggest problem with small computers in the research laboratory has been, and still is, programming. Programs are usually small to medium in size and include routines to interface with instruments, perform calculations on data, sequence through the logic of experiments, and present results. By their very nature, experiments tend to change and so the programs which perform them also must change. All parts of an experimental program are subject to change but those parts concerned with the sequencing of operations are especially so. This applies to laboratory data processing programs as well as real time control programs. Very often the experimenter making changes knows more about the experiment than the program, and is not interested in learining any more about the program than necessary. If the programming involved in an experiment becomes too troublesome then the experiment won't be done, potentially valuable research is left unfinished, and the "computer" has failed to do its job.

The threaded code technique when combined with a suitable means of compilation offers interesting possibilities for improving the laboratory computer programming situation (5). The hierarchical structure it creates is ideal for laboratory computer systems. At the bottom is the threaded program interpreter and various routines used by it coded in both microcode and assembly language. These should never be changed by the chemist type users and can be considered as part of the fixed hardware. Next come the machine code instrument interface and basic arithmetic service routines. Above them are a series of levels of threaded program building blocks defining all of the operations needed to perform experiments of a given type. At the highest level the actual experiments are defined as sequences of operations to be performed. The chemistry of an experiment is separated from the detailed computer operations so the user does not have to understand all the implementation details in order to make changes in the experiment.

Users performing experiments would confine their programming to the highest levels. With an approproate set of instructions provided by lower levels programming at the highest level should be little more than typing in sets of directives describing the conditions for specific experiments. Intermediate level threaded programs are used to implement systems for performing experiments of various types. Users working at this level must understand the chemistry involved in the experiments to be done and in addition be reasonably competent programmers. Only limited knowledge of the ultimate uses of a system is needed to implement routines at the lower levels but increased expertise in computer programming is required.

Until now the only system to make a threaded code technique directly available to the programmer has been FORTH (4). In the HISS system a general purpose macro processor is used to translate threaded code source to binary code which can be interpreted by the threaded code interpreter. This approach was used for two reasons. First, it fits the structure of threaded code very well. New instruction names can easily be defined to match the threaded code instruction set avaialble to the program being compiled. And second, it provides the flexibility required to compile complex groups of programs like the HISS operating system. Although this macro processor is not as interactive as FORTH, it produces more readable programs, handles complex syntax in a more straightforward fashion, and is less prone to errors caused by symbols being referred to outside their range.

Initially, a macro is defined to generate code for each microcode and core resident service routine. A section of threaded code can then be written as a sequence of macro names in a format resembling a simple assembly language. Threaded code procedures are defined by giving macro names and code numbers to each section of threaded code. Once defined, a procedure can then be used in any other threaded program simply be mentioning its name in the same way a service routine is called. An example demonstrating the definition of a procedure and its use is shown in Figure 3. At any time during the compilation of a procedure the compiler may be switched to assemble mode to generate regular machine code for the ASOP instruction allowing additional low level service routines. More complex syntax, such as for loops and conditionals, is handled by more complicated macros.

IV. Conclusion

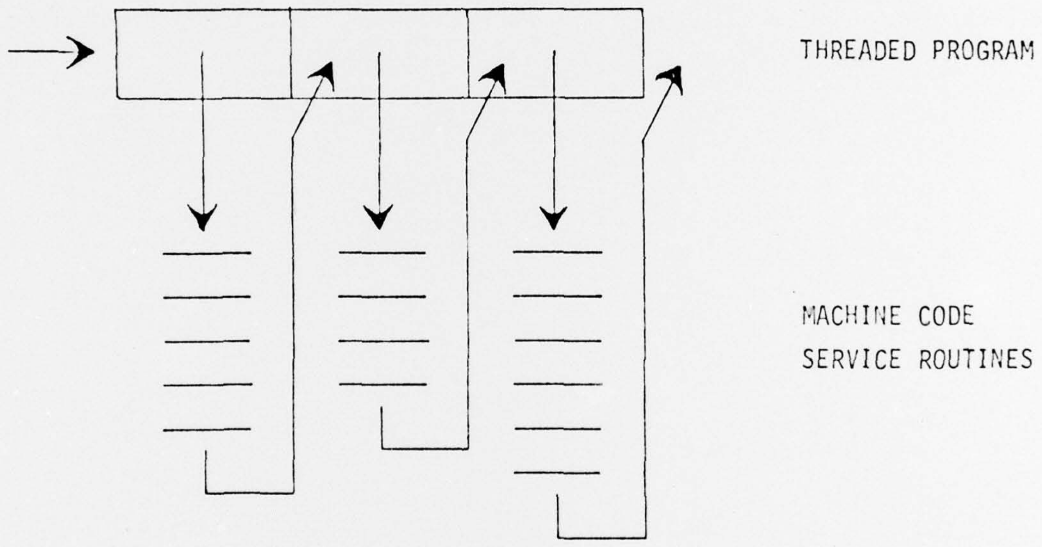
The threaded code technique in addition to saving considerable amounts of memory space with little reduction in execution speed, provides a structure which is particularly useful in programming laboratory computers. Experimental programs fit naturally into the hierarchical structure of threaded code especially when they will be continually modified. A microprogrammable computer is especially useful for implementing a threaded code system because it allows a more complex interpreter to be used without a significant penalty in execution speed. The presence of the microcoded interpreter and a machine instruction to use it transforms a typical minicomputer into a threaded code machine with interesting possibilities for structured programming. Ultimately, computer systems intended for use in laboratory environments should be specifically designed to use a hierarchical programming technique such as threaded code.

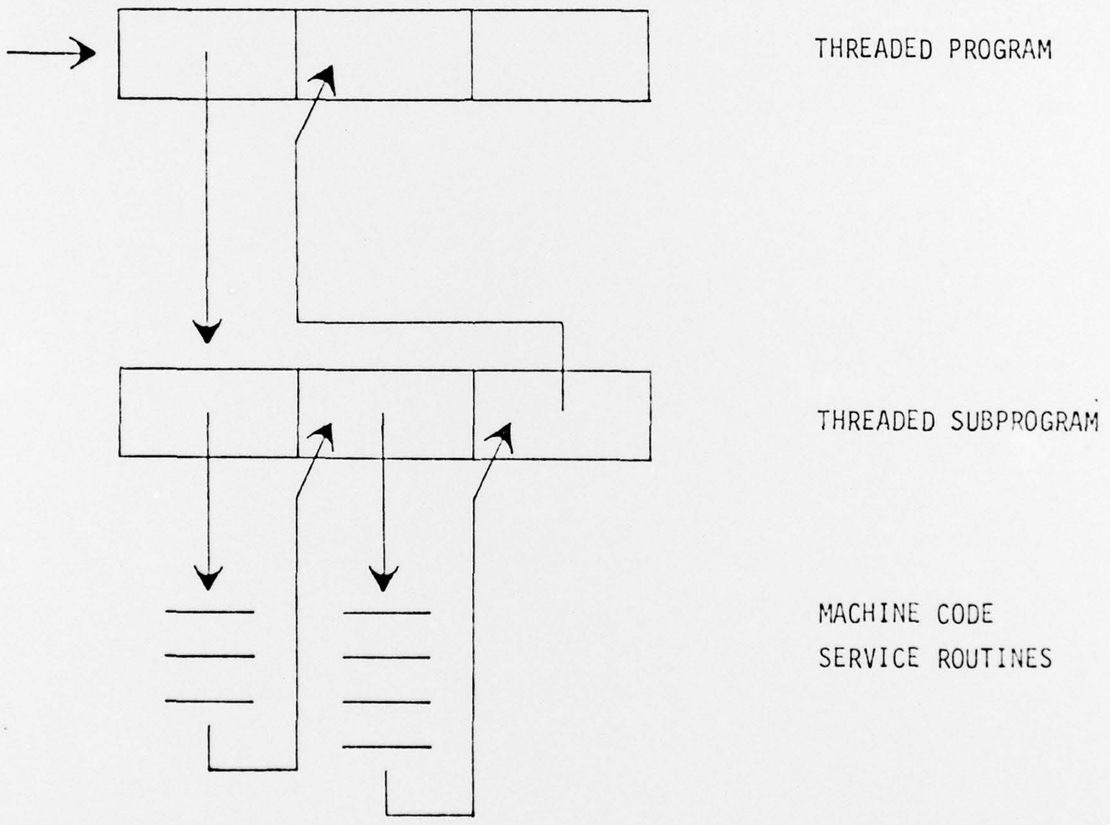
FIGURE CAPTIONS

Figure 1. Threaded code

Figure 2. Threaded code with a control stack

Figure 3. Example of the HISS system threaded code. Procedure DEL1 is defined in terms of microcoded service routines. It can then be used as an instruction in any higher level procedure such as I.MAX.


THREADED PROGRAM
MACHINE CODE
SERVICE ROUTINES


THREADED PROGRAM
THREADED SUBPROGRAM
MACHINE CODE
SERVICE ROUTINES

```
*
* PROCEDURE TO DELETE WORD 1 OF DATA STACK
*
PROC DEL1 BEGIN
XCH             * EXCHANGE TOP 2 WORDS
DEL             * DELETE TOP WORD
NEXT            * DO NEXT INSTRUCTION
END
*
* PROCEDURE TO CHOOSE THE MAXIMUM OF TWO INTEGERS
*
PROC I.MAX BEGIN
DDUP            * DUPLICATE TWO STACK WORDS
IF I.<          * IS TOP WORD SMALLER OF TWO?
THEN DEL        * YES, DELETE IT
ELSE DEL1       * NO, DELETE SECOND WORD
NEXT            * EXIT FROM PROCEDURE
END
```

References

1. Bell, J.R. Threaded code. Comm. ACM 16, 6(June 1973), 370-372.

2. Dahl, O.J., Dijkstra, E.W., and Hoare, C.A.R. Structured Programming Academic Press, London, 1972.

3. Dewar, R.B.K. Indirect threaded code. Comm. ACM 18, 6(June 1975), 330-331.

4. Moore, C.H. FORTH: a new way to program a mini-computer. Astron. Astrophys. Suppl. 15, (1974), 497-511.

5. Phillips, J.B. and Burke, M.F. Programming techniques for chromatographic experiments. J. Chrom. Sci. 14, 6(June 1976), 270-274.

6. Stoy, J.E. and Strachery, C. OS6-An experimental operating system for a small computer. Part I: General principles and structure. Computer J. 15, 2(May 1972), 117-124.

7. Wright, S. Invocation - the key to program structure. Comm. ACM 19, 6(1976), 361.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS ~~BEFORE COMPLETING FORM~~
1. REPORT NUMBER No. 8 (9) Technical	2. GOVT ACCESSION NO. rept. no. 8,	3. RECIPIENT'S CATALOG NUMBER 1 Feb 74 - 1 Feb 77.
4. TITLE (and Subtitle) (6) THREADED CODE FOR LABORATORY COMPUTERS.		5. TYPE OF REPORT & PERIOD COVERED Technical Report 2-1-74 - 2-1-77
		6. PERFORMING ORG. REPORT NUMBER Tech. Report No. 8
7. AUTHOR(s) (10) J.B./Phillips, M.F./Burke ~~and~~ G.S./Wilson		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0512
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Chemistry University of Arizona Tucson, Arizona 85721 (11) 7 Feb 77		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR051-518
11. CONTROLLING OFFICE NAME AND ADDRESS Materials Sciences Division Office of Naval Research Arlington, VA 22217 (12) 17p.		12. REPORT DATE February 7, 1977
		13. NUMBER OF PAGES 9
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research (CODE 472) Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited.

(15) N00014-75-C-0512

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

interpreter, threaded code, machine code, subroutine calls, time tradeoff, space tradeoff, compiled code, code generation, laboratory computers, problem oriented languages

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A typical minicomputer has been transformed into a threaded code machine by the addition of a simple, fast interpreter implemented in micro-code. When made directly available to the programmer, the threaded code programming technique is a very convenient and efficient means of structuring programs, particularly in systems where programs are continually being modified.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102 LF 014 6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

033 860 4/B